ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 1, January 2014

Implementation of Link Layer of USB 3.1

[1]Durgesh Kumar Sahu, [2] Prof. Shweta Singh, [3]Asst. Prof. Nitin Meena

Department of ECE, IES College of Technology, Bhopal, India

Abstract: USB is most popular interface for external computer peripherals. Peripheral devices requires very high speed data transfer rate and implements all the function of USB interface as dedicated hardware. It is standard for wired connection between two electronic devices (with maximum speed of 10Gbps), including a mobile phone and desktop computer. Different USB standard available till date are USB 1.0, USB 1.1, USB 2.0, USB 3.0 and USB 3.1. USB 3.1 is also called Enhanced SuperSpeed USB. In these standards the transmission, reception and flow control of packets on the bus is managed by the USB Host Controller. One of the essential components of USB Host Controller is USB Link Layer. This paper proposes an implementation based approach to understand the USB 3.1 standard and to develop the architecture of Link Layer in verilog and also explore the concept of CRC to achieve error free transmission.

Keywords— USB3.0,USB3.1 Universal Serial Bus, CRC in USB, Link Layer of USB 3.0 & 3.1, SuperSpeed USB, SuperSpeedPlus USB

INTRODUCTION

The Enhanced SuperSpeed USB consists of SuperSpeed USB operating at the Gen 1 speed, and SuperSpeedPlus USB operating at the Gen 2 speed. The link layer of the Enhanced SuperSpeed USB has the responsibility of maintaining the link connectivity so that successful data transfers between the two link partners are ensured. A robust link flow control is defined based on packets and link commands. Packets are prepared in the link layer to carry data and different information between the host and a device. Link commands are defined for communications between the two link partners. Packet frame ordered sets and link command ordered sets are also constructed such that they are tolerant to one symbol error. In addition, error detection capabilities are also incorporated into a packet and a link command to verify packet and link command integrity. The link layer also facilitates link training, link testing/debugging, and link power management. This is accomplished by the introduction of Link Training Status State Machine (LTSSM).

Data transfers between the two link partners are carried out using the form of a packet. A set of link commands is defined to ensure the successful packet flow across the link. Other link commands are also defined to manage the link connectivity. When symbol errors occur on the link, the integrity of a packet or a link command can be compromised. Therefore, not only a packet or link command needs to be constructed to increase the error tolerance, but the link data integrity handling also needs to be specified such that any errors that will invalidate or corrupt a packet or a link command can be detected and a link error can be recovered. There are various types of errors at the link layer. This includes an error on a packet or a link command, or an error during the link training process, or an error when a link is in transition from one state to another.

Introduction to Enhanced SuperSpeed Packets:

Enhanced SuperSpeed packets start with a 16-byte header. Some packets consist of a header only.All headers begin with the Packet Type information used to decide how to handle the packet. The header is protected by a 16-bit CRC (CRC-16) and ends with a 2-byte link control word. Depending on the Type, most packets contain routing information (Route String) and a device address triple {device address, endpoint number, and direction}. The Route String is used to direct packets sent by the host on a directed path through the topology. Packets sent by the device are implicitly routed as the hub always forwards a packet seen on any downstream port to its upstream port. There are four basic types of packets: Link Management Packets, Transaction Packets, Data Packets, and Isochronous Timestamp Packets.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 1, January 2014

A Link Management Packet (LMP) only traverses a pair of directly connected ports and is primarily used to manage that link.

A Transaction Packet (TP) traverses all the links in the path directly connecting the host and a device. It is used to control the flow of data packets, configure devices and hubs, etc. Note that a Transaction Packet does not have a data payload.

A Data Packet (DP) traverses all the links in the path directly connecting the host and a device. Data Packet consist of two parts: a Data Packet Header (DPH) which is similar to a TP and a Data Packet Payload (DPP) which consists of the data block plus a 32-bit CRC (CRC-32) used to ensure the data's integrity. An Isochronous Timestamp Packet (ITP) is a multicast packet sent by an Enhanced SuperSpeed host/hub to all active links. SuperSpeed Header and Data packet is shown in Figure: 1 & Figure: 2.

Error Detection:

The USB Enhanced SuperSpeed physical layer bit error rate is expected to be less than one in 10^{12} bits. To provide protection against occasional bit errors, packet framing and link commands have sufficient redundancy to tolerate single-bit errors. Each packet includes a CRC to provide error detection of multiple bit errors. When data integrity is required an error recovery procedure may be invoked in hardware or software. The protocol includes separate CRCs for headers and data packet payloads. Additionally, the link control word (in each packet header) has its own CRC. A failed CRC in the header or link control word is considered a serious error which will result in a link level retry to recover from the error. A failed CRC in a data packet payload is considered to indicate corrupted data and can be handled by the protocol layer with a request to resend the data packet. The link and physical layers work together to provide reliable packet header transmission. The physical layer provides an error rate that does not exceed (on average) one bit error in every 10 bits. The link layer uses error checking to catch errors and retransmission of the packet header 12 further reducing the packet header error rate.

INTRODUCTION TO CRC

The cyclic redundancy check, or CRC, is a technique for detecting errors in digital data, but not for making corrections when errors are detected. It is used primarily in data transmission. In the CRC method, a certain number of check bits, often called a checksum, are appended to the message being transmitted. The receiver can determine whether or not the check bits agree with the data, to ascertain with a certain degree of probability whether or not an error occurred in transmission. If an error occurred, the receiver sends a "negative acknowledgement" (NAK) back to the sender, requesting that the message be retransmitted. The technique is also sometimes applied to data storage devices, such as a disk drive. In this situation each block on the disk would have check bits, and the hardware might automatically initiate a reread of the block when an error is detected, or it might report the error to software. The material that follows speaks in terms of a "sender" and a "receiver" of a "message," but it should be understood that it applies to storage writing and reading as well.

I. CRC In USB 3.1

The USB spec lists two generator polynomials - one for tokens and the other for data packets. The generator polynomial for tokens is 00101, the generator polynomial for data packets is 100Bh and the generator polynomial for data payload is 04C11DB7h (Figure: 3). Since the remainder is always of smaller degree than the generator polynomial, the token CRC is a 5 bit pattern and the data CRC is a 16 bit pattern.

CRC for Link Control word

CRC-5 protects the data integrity of the Link Control Word. The implementation of CRC-5 is defined below:

- The CRC-5 polynomial shall be 00101b.
- The Initial value for the CRC-5 shall be 11111b.
- CRC-5 is calculated for the remaining 11 bits of the Link Control Word.
- CRC-5 calculation shall begin at bit 0 and proceed to bit 10.
- The remainder of CRC-5 shall be complemented, with the MSb mapped to bit 11, the next MSb mapped to bit 12, and so on, until the LSb mapped to bit 15 of the Link Control Word.
- The residual of CRC-5 shall be 01100b.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 1, January 2014

Note: The inversion of the CRC-5 remainder adds an offset of 11111b that will create a constant CRC-5 residual of 01100b at the receiver side.

CRC for Packet Header

The implementation of CRC-16 on the packet header is defined below:

• The polynomial for CRC-16 shall be 100Bh.

Note: The CRC-16 polynomial is not the same as the one used for USB 2.0.

• The initial value of CRC-16 shall be FFFFh.

• CRC-16 shall be calculated for all 12 bytes of the header information, not inclusive of any packet framing symbols.

• CRC-16 calculation shall begin at byte 0, bit 0 and continue to bit 7 of each of the 12 bytes.

• The remainder of CRC-16 shall be complemented.

• The residual of CRC-16 shall be F6AAh.

Note: The inversion of the CRC-16 remainder adds an offset of FFFFh that will create a constant CRC-16 residual of F6AAh at the receiver side.

CRC for Data Packet Payload

The DPP section consists of 0 to 1024 data bytes followed by 4 bytes CRC-32. Any premature termination of a DPP shall contain a DPPABORT ordered set. A DPP shall immediately follow its corresponding DPH with no spacing in between. CRC-32 protects the data integrity of the data payload. CRC-32 is as follows:

• The CRC-32 polynomial shall be 04C1 1DB7h.

• The CRC-32 Initial value shall be FFFF FFFFh.

• CRC-32 shall be calculated for all bytes of the DPP, not inclusive of any packet framing symbols.

• CRC-32 calculation shall begin at byte 0, bit 0 and continue to bit 7 of each of the bytes of the DPP.

• The remainder of CRC-32 shall be complemented.

• The residual of CRC-32 shall be C704DD7Bh.

Note: The inversion of the CRC-32 remainder adds an offset of FFFF FFFFh that will create a constant CRC-32 residual of C704DD7Bh at the receiver side.

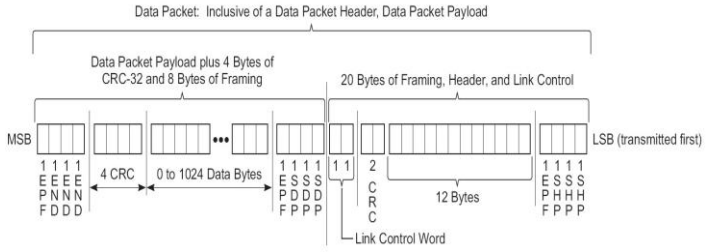

Figure: 1 SuperSpeed DP Format

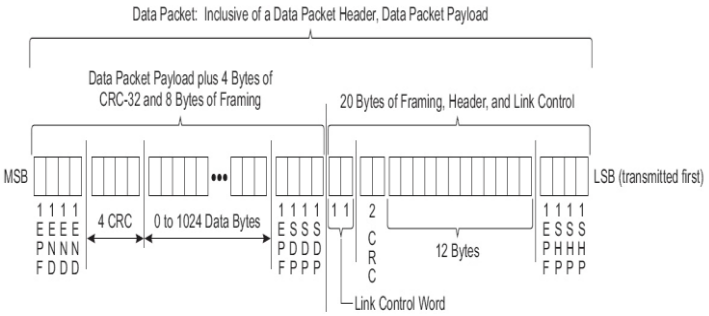

Figure : 2 SuperSpeedPlus DP Format

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 1, January 2014

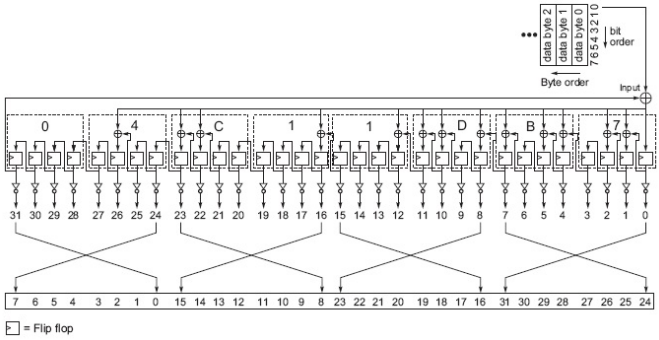

Figure: 3 CRC32 Generator

II. Proposed Implementation Method

An intuitive way to generate the CRC for an input pattern would be to simply divide this pattern by the generator polynomial. The division can use the same approach used in integer division. If the Most Significant digit/Bit (MSB) of the dividend is a 1, the divisor is subtracted from the dividend to generate a new dividend. In modulo 2 arithmetic, this subtraction is simply a bit-wise XOR. The same step is repeated for the next MSB in sequence through all the bits including the Least Significant Bit (LSB). The remainder at this point is the remainder from dividing the input pattern multiplied by x by the generator Polynomial of degree d (d is 5 for token CRC and 16 for data CRC). The implementation called for in the USB spec differs from the above intuitive approach in three ways which are mathematically insignificant.

1. The implementation starts off with the shift register loaded with all 1s. Without this, leading 0s in front of a packet would not be protected by the CRC generated. Mathematically this is equivalent to adding a scaled constant to the dividend. The scaling is a function of number of bits in the input pattern. At the receiving target, the shift register is likewise primed with 1s. This ensures that the same scaled constant is added to the dividend at the target (assuming no bits are lost in transit). So if no bits are corrupted in transit, the same remainder will be generated at both ends. In equation form the scaling creates the dividend D(x) = x F(x) + x L(x) where F(x) is a degree (k-1) polynomial 32k representing the k bits of the data stream and L(x) is a degree (d-1) polynomial with all coefficients equal to one and d is degree of the generator polynomial.
2. The implementation uses commutability and associability of the XOR operation to advantage. In the intuitive approach, the new dividend is derived by subtracting the divisor from the dividend. In the implementation, this subtraction is done bit by bit on the dividend by accumulating and shifting the remainder.
3. The remainder is bit-wise inverted before being appended as the checksum to the input pattern. Without this modification, trailing zeros at the end of a packet could not be detected as data transmission errors. Mathematically, this is equivalent to adding a known constant to the remainder. Again this is mathematically insignificant to the operation of CRC. In equation form, CRC= L(x) + R(x) where R(x) is remainder obtained by dividing D(x) by the generator polynomial G(x).
4. Checking the CRC at the target is the same as generating the CRC on an input pattern which now consists of the original input pattern followed by the inverted remainder. Mathematically, this new polynomial should be perfectly divisible by the generator polynomial except for the residual due to the known constant discussed in item 3 above. (This can be intuitively understood by realizing that the appending of the remainder to the LSB of the dividend is equivalent to subtracting it from the old dividend). In equation form, the transmitted and received data is M(x) = x F(x) + CRC. When CRC is 32generated on this pattern M(x), the remainder R'(x) is x L(x)/G(x) and can be derived 32from the above equations and some properties of modulo 2 arithmetic. R'(x) is a unique polynomial (i.e. coefficients are always the same) since L(x) and G(x) are unique. R'(x) is termed

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 1, January 2014

the residue or residual polynomial. For the token generator polynomial, the residual is 01100; for the data CRC polynomial the residual is 1000000000001101

III. CONCLUSION & FUTURE SCOPE

The main outcome is in the form of better understanding about which strategy is to be applied for Communication in USB serial transfer. After the study it can be said that the communication in Enhanced SuperSpeed USB is more faster and secure using CRC than previous USB versions. The best suited technique for applying CRC is with Linear Feedback Shift Register. For reliable communication USB use Dual Port Asynchronous RAM and Synchronous FIFO to store latest transmit Frames(at least last four) so that in case of receiving Bad Frame detect by Receiver can be retransmit. USB is best mean to expand Memory and to connect Peripheral or Devices to the system as there is no requirement of Rebooting the system.

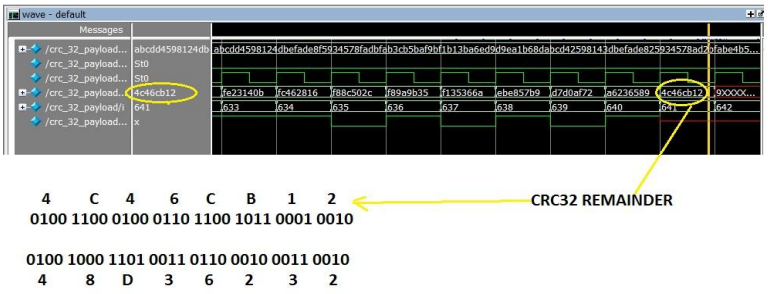

Figure: - 4 Generation of CRC32 at Transmitter side

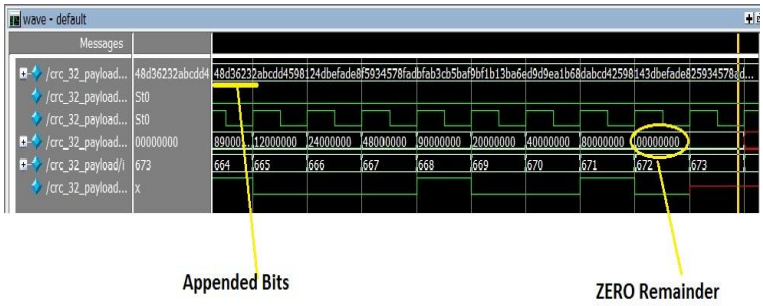

Figure: - 5 Checking of Received Data Payload with CRC32

REFERENCES

[1] USB 3.1 Specification Revision 1.0, http://www.usb.org, 26 July 2013.
[2] 2013 IEEE Symposium on Design and Diagnostics of Electronics Circuits and Systems, "A System Level Overview and Comparison of Three High-Speed Serial Links: USB 3.0, PCI Express 2.0 and LLI 1.0".
[3] IEEE 2012- 9th International Multi-Conference on System, Signal and Devices "15 Gbps Communication over an USB3.0Cable and Even More", Dr.Andreas C. Wolf & Dr. Christoph Scheytt.
[4] Hardware/Software Co-design in USB 3.0 Mass Storage Applications, Ireneusz Sobański & Wojciech Sakowski, ICSES 2010- The International Conference on Signal & Electronic Systems, Gliwice, Poland 2010.
[5] Modelsim Tutorial, Software version 6.6d, May 2010.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 1, January 2014

[6] G.M. Bhat, M. Mustafa, Shabir Ahmad and Javaid Ahmad, " VHDL modeling and simulation of data scrambler and descrambler for secure data communication ",IJST, Vol.2, No. 10, October 2009
[7] USB 3.0 SuperSpeed CDR Model whitepaper, Reversion 0.5, January 2009.
[8] USB 3.0 Specification Revision 1.0, http://www.usb.org, November 2008.
[9] Ryan W. Apperson, Zhiyi Yu, Michael J. Meeuwsen, Tinoosh Mohsenin, and Bevan M. Baas, "A Scalable Dual-Clock FIFO for Data Transfers Between Arbitrary and Haltable Clock Domains", IEEE, VOL. 15, NO. 10, OCTOBER 2007.
[10] Tobias Dubois, Mohamed Azimane, Erik Larsson, Erik Jan Marinissen, Paul Wielage and Clemens Wouters, "Test Quality Analysis and Improvement for an Embedded Asynchronous FIFO", NXP Semiconductors, EDAA, 2007.
[11] Xin Wang and Jari Nurmi, "A RTL Asynchronous FIFO Design Using Modified Micropipeline", IEEE, 2006.
[12] Ellis Horowitz & Sartaj Sahni ,"Fundamentals Of Data Structures", March 2004.
[13] Chris Borrelli,"IEEE 802.3 Cyclic Redundancy Check" , March 2001.
[14] USB 2.0 Specification Revision 2.0, http://www.usb.org, April 27, 2000.
[15] Nova," Linear Feedback Shift Register Megafunction", Nova Engineering, ver. 1, December 1996.
[16] Duolos, "The Verilog Golden Reference Guide", version 1.0, August 1996.